| | | |
|---|---|---|
| Title | : | Removing Overflow Rows in a Relational Database |
| Inventors | : | Stephen R. Cole and Michael J. McLaughlin |
| Serial No. | : | 10/757,741 |
| Filed | : | 22 Oct 2002 |
| Examiner | : | Cheyne D. Ly |
| Art Unit | : | 2168 |
| Docket | : | 149-0168US |
| Customer | : | 29855 |

**Mail Stop Appeal Brief Patents**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

## APPEAL BRIEF

This is an appeal from the rejection of claims 1-72 in the Final Office Action dated 01 NOV 2007.

## Table of Contents

## I. REAL PARTY IN INTEREST

BMC Software, Inc. of Houston, Texas is the real party in interest.

## II. RELATED APPEALS AND INTERFERENCES

None.

## III. STATUS OF CLAIMS

Claims 1-72 are rejected. Claims 1-72 are appealed.

## IV. STATUS OF AMENDMENTS

None filed.

## V. SUMMARY OF CLAIMED SUBJECT MATTER

This section provides a concise explanation of the subject matter defined in each independent claim involved in the appeal, referring to the specification and to the drawings as required by 37 C.F.R. 41.37(c)(l)(v). Note that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element.

### A. Concise Explanation of Independent Claim 1

Independent claim 1 is directed to an overflow row repair method, comprising: retrieving a page of memory associated with a source table (¶ [0018], ¶¶ [0020]-[0021], 400 in Fig. 4, and 505 in Fig. 5); interrogating the page of memory to identify an overflow row (¶ [0018], ¶¶ [0020]-[0021], 400 in Fig. 4, and 505 in Fig. 5); unloading

only the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 205 in Fig. 2, and 425 in Figs. 4-5); deleting the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 215 in Fig. 2, and 440 in Figs. 4-5); and loading the previously unloaded identified overflow row into the source table (¶ [0016], ¶ [0018], ¶ [0020], 220 in Fig. 2; and 445 in Figs. 4-5).

**B. Concise Explanation of Independent Claim 17**

Independent claim 17 is directed to an overflow row repair method, comprising: retrieving one or more pages of memory associated with a source table (¶ [0018], ¶¶ [0020]-[0021], 400 in Fig. 4, and 505 in Fig. 5); interrogating the one or more pages of memory to identify one or more overflow rows (¶ [0016], ¶ [0018], ¶¶ [0020]-[0021], 205 in Fig. 2, 400 in Fig. 4, and 505 in Fig. 5); unloading only a first portion of the identified overflow rows from the source table (¶ [0016], ¶ [0018], ¶ [0020], 210 in Fig. 2, and 425 in Figs. 4-5); deleting the unloaded overflow rows from the source table (¶ [0016], ¶ [0018], ¶ [0020], 215 in Fig. 2, and 440 in Figs. 4-5); and reloading the previously unloaded overflow rows into the source table (¶ [0016], ¶ [0018], ¶ [0020], 220 in Fig. 2; and 445 in Figs. 4-5).

**C. Concise Explanation of Independent Claim 28**

Independent claim 28 is directed to an overflow row repair method, comprising: identifying an overflow row associated with a source table from a non-source table data source (¶ [0016], ¶ [0018], ¶¶ [0020]-[0021], 205 in Fig. 2, 400 in Fig. 4, and 505 in Fig. 5); unloading only the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 210 in Fig. 2, and 425 in Figs. 4-5); deleting the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 215 in Fig. 2, and 440 in Figs. 4-5); and reloading the previously deleted identified overflow row into the source table (¶ [0016], ¶ [0018], ¶ [0020], 220 in Fig. 2; and 445 in Figs. 4-5).

**D. Concise Explanation of Independent Claim 43**

Independent claim 43 is directed to a program storage device, readable by a programmable control device, comprising instructions stored on the program storage

device for causing the programmable control device to: retrieve a page of memory associated with a source table (¶ [0018], ¶¶ [0020]-[0021], 400 in Fig. 4, and 505 in Fig. 5); interrogate the page of memory to identify an overflow row (¶ [0016], ¶ [0018], ¶¶ [0020]-[0021], 205 in Fig. 2, 400 in Fig. 4, and 505 in Fig. 5); unload only the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 210 in Fig. 2, and 425 in Figs. 4-5); delete the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 215 in Fig. 2, and 440 in Figs. 4-5); and load the previously unloaded identified overflow row into the source table (¶ [0016], ¶ [0018], ¶ [0020], 220 in Fig. 2; and 445 in Figs. 4-5).

**E. Concise Explanation of Independent Claim 58**

Independent claim 58 is directed to a program storage device, readable by a programmable control device, comprising instructions stored on the program storage device for causing the programmable control device to: identify an overflow row associated with a source table from a non-source table data source (¶ [0016], ¶ [0018], ¶¶ [0020]-[0021], 205 in Fig. 2, 400 in Fig. 4, and 505 in Fig. 5); unload only the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 210 in Fig. 2, and 425 in Figs. 4-5); delete the identified overflow row from the source table (¶ [0016], ¶ [0018], ¶ [0020], 215 in Fig. 2, and 440 in Figs. 4-5); and reload the previously deleted identified overflow row into the source table (¶ [0016], ¶ [0018], ¶ [0020], 220 in Fig. 2; and 445 in Figs. 4-5).

## VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

**A.** Claims 1-6, 14-20, 27-30, 39-48, 55-60, and 69-72 stand rejected under 35 U.S.C. 102(b) as allegedly being anticipated by the publication entitled "A method for on-line reorganization of a database" by G. H. Sockut, IBM Systems Journal, Vol. 36, No. 3, 1979 (hereinafter "Sockut").

**B.** Claims 7-13, 21-26, 31-38, 49-54, and 61-68 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Sockut as applied to claims 1-6, 14-20, 27-30, 39-48, 55-60 and 69-72 and further in view of one or more secondary references.

## VII. ARGUMENT

The claims do not stand or fall together. Instead, appellants present separate arguments for various independent and dependent claims. Each of these arguments is separately presented below under separate headings and sub-heading as required by 37 C.F.R. 41.37(c)(l)(vii).

### A. Sockut Fails to Anticipate Claims 1-6, 14-20, 27-30, 39-48, 55-60, and 69-72 under 35 U.S.C. § 102(b)

Claims 1-6, 14-20, 27-30, 39-48, 55-60 and 69-72 stand rejected as allegedly being anticipated under 35 U.S.C. 102(b) by Sockut.

1. Independent Claims 1, 17, 28, 43, and 58

Sockut fails to teach or suggest each and every claimed element of independent claims 1, 17, 28, 43, and 58 because (1) Sockut fails to disclose unloading ***only*** identified overflow rows from a source table, (2) Sockut fails to disclose deleting identified overflow rows from a source table and loading previously unloaded overflow rows back into the same source table, and (3) the Examiner's interpretation of the claims is improperly inconsistent with Appellant's specification.

(a) *Rejection*

The Examiner rejects claim 1 as follows:

> In regard to claim 1, Sockut discloses an overflow row repair method, comprising: Retrieving a page of memory associated with a source table (page 4, paragraph 7, especially, "the table space or partition on which reorganization operates", and page 11, lines 1-39, especially, "scanning the file pages ..."); Interrogating the page of memory to identify an overflow row (page 11, lines 1-39, especially, "we find a ... overflow record"); Unloading the identified overflow row from the source table (page 11, lines 1-39, especially, "we find...an overflow record...Unloading an overflow..."); Deleting the identified overflow row from the source table (page 1, Abstract etc., and page 4, 6th paragraph, especially, "removes overflow ...", and pages 9-10, Table 2); and Loading the previously unloaded identified overflow row into the source table (page 11, lines 1-39, especially, "Reloading of data ...").

Final Office Action dated 01 NOV 2007 at pg. 3-4, ¶ 9.

The Examiner asserts the same rejection against independent claims 17, 28, 43 and 58. *See* Final Office Action at pg. 5, ¶¶ 15-16.

(b) *Discussion of Sockut*

Sockut apparently describes a reorganization technique that, as an improvement, allows for reorganization online. In Sockut's reorganization technique, an "area being reorganized is unloaded." Sockut at 5 (1st full ¶).[1] See *also*, *Id.* at 1 (Abstract) & 2 (2nd ¶). Sockut uses "the term area being reorganized (often shortened to just area) to mean the table space or partition on which reorganization operates." Sockut at page 4, 7th ¶ (emphasis added).[2] In line with this definition, Sockut explicitly and repeatedly describes unloading an entire tablespace or partition *en masse* into an unload file. *See e.g.*, *Id.* at 1 (Abstract), 2 (2nd ¶), 4 (7th ¶), 5 (1st, 3rd and 4th full ¶), 6 (1st and 7th full ¶), 10 (7th ¶), 11 (1st ¶), and Figs. 4 and 6.

To unload the "area being reorganized" (*i.e.*, tablespace or partition), Sockut scans each file page and its ID maps in the ***old copy*** of the area and unloads the data into the unload file. *Id.* at 5 (3rd ¶) & 11 (1st ¶). When scanning and unloading, Sockut describes how regular records, overflow records, and pointer records are handled when found. When regular and overflow records are found, the data is simply unloaded to the unload file. *Id.* at 11 (2nd ¶). When a pointer is found, however, it is not followed . *Id.* at 11 (3rd ¶). Subsequent to unloading, the unload file is sorted and then reloaded into a ***new copy*** of the area. *See Id.* at 11 (4th-5th ¶); *See also, Id. at* 5 (1st full ¶). Finally, "[t]he new copy is then brought up to date ... by applying log entries ... Future access by users is then switched to the new (reorganized) copy of the area." *Id.* at 5 (1st full ¶).

---

1 All references to Sockut refer to page and paragraph numbers in that copy provided to Appellant by the Examiner in the Office Action dated 28 December 2006.

2 As is common in the art, Sockut defines a table space as "a region of storage that stores the data records for one or more tables." Sockut at 2 (9th ¶). Also as common in the art, Sockut describes a partitioned table space as one in which "the table space ... [is divided] ... into partitions according to values of the indexed key ... Partitions reside in separate files, whereas a nonpartitoned table space can reside in one file." Sockut at 4 (1st ¶).

(c) *Discussion of Independent Claims*

In contrast to Sockut, independent claim 1 calls for retrieving a page of memory associated with a source table, interrogating that page to identify an overflow row, unloading only that identified overflow row from the source table, deleting that identified overflow row from the source table, and loading the previously unloaded identified overflow row into the source table. Independent claim 17 calls for similar elements as claim 1 but retrieves one or more pages of memory, identifies one or more overflow rows, and unloads only a first portion of the identified overflow rows. Independent claim 43 calls for similar elements as claim 1 but is directed to a program storage device.

Independent claim 28 calls for identifying an overflow row associated with a source table from a non-source table data source, unloading only the identified overflow row from the source table, deleting the identified overflow row from the source table, and reloading the previously deleted identified overflow row into the source table. Independent claim 58 calls for similar claimed elements as claim 28 but is directed to a program storage device.

(d) *Sockut Fails to Disclose Unloading Only Identified Overflow Rows from a Source Table*

Sockut fails to teach or fairly suggest unloading ***only*** identified overflow rows from a source table as called for in claims 1, 17, 28, 43, and 58. As noted above, Sockut teaches unloading an "area being reorganized" and specifically refers to such an area as an entire tablespace or partition on which reorganization operates. Sockut at 4 (7th ¶). Indeed, Sockut's reorganization technique explicitly and repeatedly relies upon unloading an entire tablespace or partition *en masse* from an old copy into an unload file. *See e.g., Id.* at 1 (Abstract), 2 (2nd ¶), 4 (7th ¶), 5 (1st, 3rd and 4th full ¶), 6 (1st and 7th full ¶), 10 (7th ¶), 11 (1st ¶), and Figs. 4 and 6. In fact, Sockut explicitly describes unloading both regular and overflow records to the unload file when scanning file pages (*Id.* at 11, 2nd ¶) so that Sockut does not unload only overflow rows but also unloads regular records together with overflow records. In this way, Sockut operates according to the prior art as described in Appellant's "Background" section in which prior art

overflow reformation techniques unload a complete tablespace or partition for reloading into a new copy of the tablespace or partition. *See* Specification at ¶ 6 and Fig. 1.

In contrast, Appellant's claims call for unloading ***only*** identified overflow rows from a source table. For example, independent claim 1 expressly unloads ***only*** an identified overflow row from a source table. By unloading only identified overflow rows, independent claims 1, 17, 28, 43, and 58 thereby expressly do not claim the act of unloading an entire tablespace or partition *en masse* (*i.e.,* both regular and overflow records) as disclosed by Sockut. For this reason, Sockut fails to anticipate independent claims 1, 17, 28, 43, and 58.

(e) *Sockut Fails to Disclose Deleting Overflow Rows from a Source Table and Loading Overflow Rows into the Source Table*

In addition to failing to unload ***only*** identified overflow rows from a source table, Sockut fails to teach or fairly suggest deleting such identified overflow rows from the source table and loading the previously unloaded identified overflow rows back into the same source table, as called for in Appellant's claims. For example, independent claim 1 expressly calls for deleting an identified overflow row ***from the source table*** after unloading only the identified overflow row from the source table and calls for loading the previously unloaded identified overflow row ***into the source table***.

As noted above, Sockut unloads data in *an old copy of the area to be reorganized* into an unload file and then reloads records from the unload file into *a new copy of the area*. *See* Sockut at 11. Again, Sockut operates squarely according to the prior art as described in Appellant's "Background" section in which prior art overflow reformation techniques (1) unload a complete table, (2) reload the data into a new copy of the table, and (3) replace the old table with the new table. *See* Specification at ¶ 6 and Fig. 1. By unloading only identified overflow rows in a source table, deleting them from the source table, and then loading the previously unloaded overflow rows back into the same source table, Appellant's claims expressly do not claim the act of unloading data in an old copy of an area to be reorganized and loading the unloaded data into a

new copy of the area as disclosed in Sockut. For this reason, Sockut fails to anticipate independent claims 1, 17, 28, 43, and 58.

(f) *The Examiner's Interpretation of the Claim Language is Inconsistent with Appellant's Specification*

In rejecting independent claims 1, 17, 28, 43, and 58, the Examiner has interpreted the claim language in a way inconsistent with Appellant's specification. *See* MPEP 2111 ("During patent examination, the pending claims must be 'given their broadest reasonable interpretation consistent with the specification.'"). In particular, the Examiner defines the term "only" in Appellant's independent claims 1, 17, 28, 43, and 58 as meaning "being the single one or relatively few of the kind." Final Office Action at pg. 2, ¶ 3. Based on this definition, the Examiner maintains that Sockut anticipates Appellant's claims because Sockut discloses unloading overflow records, albeit along with unloading regular records as well of an entire tablespace or partition. *See* Sockut at 11, lines 1-11. Unfortunately, it appears that the Examiner has interpreted Appellant's claims in a way inconsistent with Appellant's specification, which is improper. In particular, the Examiner's interpretation of the claim language fails to recognize that the claimed subject matter is directed to identifying, unloading, deleting, and loading "exclusively" (a synonym for only) overflow rows in the same source table. This meaning is consistent with Appellant's specification in which overflow records are exclusively identified, unloaded, deleted, and loaded in the same source table. *See e.g.*, Specification at ¶¶ [0016], [0018], [0020], and Figs. 2, 4, & 5.

By contrast, Sockut taken as a whole clearly fails to identify, unload, delete, and load only (exclusively) overflow rows in the same source table. Rather and as noted previously, Sockut unloads an old copy of an "area to be reorganized" (*i.e.,* table space or partition) *en masse* into an unload file and loads the data from the unload file into a new copy of the area. *See e.g., Id.* at 1 (Abstract), 2 (2nd ¶), 4 (7th ¶), 5 (1st, 3rd and 4th full ¶), 6 (1st and 7th full ¶), 10 (7th ¶), 11 (1st ¶), and Figs. 4 and 6. In this unloading, Sockut unloads regular and overflow records and not overflow rows

exclusively. *See Id.* at 11 (2nd ¶). For this reason, Sockut fails to anticipate independent claims 1, 17, 28, 43, and 58.

(g) *Summary*

Sockut fails to disclose unloading only (exclusively) identified overflow rows from a source table and fails to disclose deleting the identified overflow rows from the source table and loading the previously unloaded overflow rows back into the same source table. In addition, the Examiner's interpretation of Appellant's claims and is improperly inconsistent with Appellant's specification. For at least these reasons, Sockut fails to teach or suggest each and every claimed element of independent claims 1, 17, 28, 43 and 58, and Appellant respectfully requests that the Board reverse the Examiner's rejection.

2. Claims 2-6, 14-16, 18-20, 27, 29-30, 39-42, 44-48, 55-57, 59-60, and 69-72

Because claims 2-6, 14-16, 18-20, 27, 29-30, 39-42, 44-48, 55-57, 59-60, and 69-72 depend from independent claims 1, 17, 28, 43, and 58, Sockut fails to teach or suggest each and every claimed element recited in these dependent claims for at least the same reasons presented above with respect to the independent claims. Accordingly, Appellant respectfully requests that the Board reverse the Examiner's rejection of dependent claims 2-6, 14-16, 18-20, 27, 29-30, 39-42, 44-48, 55-57, 59-60, and 69-72.

**B. Sockut and Secondary References Fail to Render Claims 7-13, 21-26, 31-38, 49-54, and 61-68 as Obvious**

1. Claims 7, 21, 31, 32, 49, 61, and 62

Claims 7, 21, 31, 32, 49, 61, and 62 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Sockut as applied to claims 1-6, 14-20, 27-30, 39-48, 55-60 and 69-72. Each of these claims depends from one of Appellant's independent claims 1, 17, 28, 43, and 58. For at least the same reasons presented above with respect to the independent claims, Sockut fails to teach or suggest each claim element of these

dependent claims, and Appellant respectfully requests that the Board reverse the Examiner's rejection of claims 7, 21, 31, 32, 49, 61, and 62.

2. Claims 8-13, 22-26, 33-38, 50-54, and 63-68

Claims 8-9, 11-12, 22, 24-26, 33-34, 37-38, 50-51, 53-54, 63-64, and 67-68 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Sockut in view of US 5,899,993 ("Jenkins"), claims 10, 23, 36, 52, and 66 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Sockut in view of Jenkins and further in view of US 2005/0080979 ("Wu"), and claims 13, 35, and 65 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Sockut in view of Jenkins and further in view of US 6,047,285 ("Jacobs")

Each of the above claims depends from one of the independent claims 1, 17, 28, 43, and 58. As noted above, Sockut fails to teach or suggest each and every claimed element of these independent claims, thereby failing to disclose each and every claimed element of these dependent claims.

The secondary references of Jenkins, Wu, and Jacobs fail to provide the limitations missing from Sockut. In particular, Jenkins discloses a method and apparatus for enabling a constraint without prohibiting updates to constrained data during validation. *See e.g.,* Jenkins at col. 3, lines 36-38. Wu discloses a way to maximize the utility of allocation of resources without needing to know the utility as a function of those allocated resources. *See e.g.,* Wu at ¶ [0011]. Jacobs discloses a uniqueness-required index and a corresponding non-uniqueness count to support enforcing deferred uniqueness constraints in a database. *See e.g.,* Jacobs at col. 4, lines 65-67. Each of these references fails to identify, unload, delete, and load only (exclusively) overflow rows in the same source table. For at least these reasons, Sockut in view of these secondary references fails to render the dependent claims obvious, and Appellant respectfully requests that the Board reverse the Examiner's rejection of these dependent claims.

### C. Final Remarks

Because Sockut fails to teach or suggest *each and every* claimed element recited in Appellant's independent claims 1, 17, 28, 43, and 58, Sockut fails to anticipate these claims. For at least these same reasons, dependent claims 2-16, 18-27, 29-42, 44-57, and 59-72 are not anticipated by Sockut nor rendered obvious over Sockut in view of the secondary references cited. Consequently, Appellant respectfully requests that the Board grant Appellant's appeal and reverse the Examiner's rejections of claims 1-72.

Respectfully submitted,

**/Sean McDermott/**
Reg. No. 49,000

*Wong, Cabello, Lutsch, Rutherford & Brucculeri, L.L.P.*
Customer No. 29855
20333 SH 249, Suite 600
Houston, Texas 77070
Voice: 832-446-2416
Facsimile: 832-446-2424

## VIII. CLAIMS APPENDIX

1. (Previously Presented) An overflow row repair method, comprising:
retrieving a page of memory associated with a source table;
interrogating the page of memory to identify an overflow row;
unloading only the identified overflow row from the source table;
deleting the identified overflow row from the source table; and
loading the previously unloaded identified overflow row into the source table.

2. (Original) The method of claim 1, wherein the acts of retrieving and interrogating are repeated for each page of memory comprising the source table.

3. (Original) The method of claim 1, wherein the acts of retrieving and interrogating are repeated for less than all pages of memory comprising the source table.

4. (Original) The method of claim 1, wherein the source table further comprises an index.

5. (Original) The method of claim 1, wherein the act of retrieving comprises retrieving the page of memory from a buffer pool.

6. (Original) The method of claim 1, wherein the act of retrieving comprising retrieving the page of memory from a direct access storage device.

7. (Original) The method of claim 1, further comprising locking the source table before deleting the identified overflow row from the source table.

8. (Original) The method of claim 7, wherein the act of deleting comprises:
identifying a constraint associated with the identified overflow row;
disabling the identified constraint; and
deleting the identified overflow row from the source table.

9. (Original) The method of claim 8, further comprising locking a table associated with the identified constraint prior to the act of disabling.

10. (Original) The method of claim 8, wherein the act of disabling comprises dropping the identified constraint.

11. (Original) The method of claim 8, wherein the act of loading comprises:
inserting the previously deleted identified overflow row into the source table;
restoring the identified constraint; and
unlocking the source table.

12. (Original) The method of claim 11, wherein the act of unlocking the source table unlocks a table associated with the identified constraint.

13. (Original) The method of claim 9, wherein if the identified constraint is a deferred constraint, the table associated with the identified constraint is not locked.

14. (Original) The method of claim 1, wherein the act of unloading comprises executing a structured query language SELECT statement.

15. (Original) The method of claim 14, wherein the act of deleting comprises executing a structured query language DELETE statement.

16. (Original) The method of claim 14, wherein the act of loading comprises executing a structured query language INSERT statement.

17. (Previously Presented) An overflow row repair method, comprising:
retrieving one or more pages of memory associated with a source table;
interrogating the one or more pages of memory to identify one or more overflow rows;
unloading only a first portion of the identified overflow rows from the source table;
deleting the unloaded overflow rows from the source table; and
reloading the previously unloaded overflow rows into the source table.

18. (Original) The method of claim 17, wherein the act of retrieving comprises retrieving from a buffer pool.

19. (Original) The method of claim 17, wherein the act of retrieving comprises retrieving from a direct access storage device.

20. (Original) The method of claim 17, wherein the act of retrieving comprises retrieving one or more data pages associated with the source table.

21. (Original) The method of claim 17, further comprising locking the source table prior to the act of deleting.

22. (Original) The method of claim 21, further comprising:
identifying a constraint associated with at least one of the identified overflow rows; and
disabling the identified constraint.

23. (Original) The method of claim 22, wherein the act of disabling comprises dropping the identified constraint.

24. (Original) The method of claim 22, further comprising locking a table associated with the identified constraint prior to disabling the identified constraint.

25. (Original) The method of claim 22, wherein the act of reloading comprises:
inserting the previously unloaded overflow rows into the source table;
restoring the identified constraint; and
unlocking the source table and the table associated with the identified constraint.

26. (Original) The method of claim 25, wherein the act of restoring comprises rebuilding the identified constraint.

27. (Original) The method of claim 17, wherein the acts of unloading, deleting and reloading are repeated, wherein each iteration unloads, deletes and reloads a portion of the identified overflow rows.

28. (Previously Presented) An overflow row repair method, comprising:
identifying an overflow row associated with a source table from a non-source table data source;
unloading only the identified overflow row from the source table;
deleting the identified overflow row from the source table; and
reloading the previously deleted identified overflow row into the source table.

29. (Original) The method of claim 28, wherein the non-source table data source comprises a database log file.

30. (Original) The method of claim 28, wherein the source table further comprises an index.

31. (Original) The method of claim 28, further comprises locking the source table before deleting the identified overflow row from the source table.

32. (Original) The method of claim 31, further comprises unlocking the source table after the act of reloading the identified overflow row.

33. (Original) The method of claim 31, wherein the act of deleting comprises:
identifying a constraint associated with the identified overflow row;
disabling the identified constraint; and
deleting the identified overflow row from the source table.

34. (Original) The method of claim 33, wherein the act of disabling comprises locking a table associated with the identified constraint prior to the act of disabling the identified constraint.

35. (Original) The method of claim 34, wherein if the identified constraint is a deferred constraint, the table associated with the identified constraint is not locked.

36. (Original) The method of claim 33, wherein the act of disabling comprises deleting the identified constraint.

37. (Original) The method of claim 33, wherein the act of reloading comprises:
inserting the previously unloaded identified overflow rows into the source table;
restoring the identified constraint;
unlocking the source table.

38. (Original) The method of claim 37, wherein the act of restoring comprises rebuilding the identified constraint.

39. (Original) The method of claim 28, wherein the acts of unloading, deleting and loading are performed on a plurality of identified overflow rows at a time.

40. (Original) The method of claim 28, wherein the act of unloading comprises executing a structured query language SELECT statement.

41. (Original) The method of claim 40, wherein the act of deleting comprises executing a structured query language DELETE statement.

42. (Original) The method of claim 41, wherein the act of reloading comprises executing a structured query language INSERT statement.

43. (Previously Presented) A program storage device, readable by a programmable control device, comprising instructions stored on the program storage device for causing the programmable control device to:

retrieve a page of memory associated with a source table;
interrogate the page of memory to identify an overflow row;
unload only the identified overflow row from the source table;
delete the identified overflow row from the source table; and
load the previously unloaded identified overflow row into the source table.

44. (Original) The program storage device of claim 43, wherein the instructions to retrieve and interrogate are repeated for each page of memory comprising the source table.

45. (Original) The program storage device of claim 43, wherein the instructions to retrieve and interrogate are repeated for less than all pages of memory comprising the source table.

46. (Original) The program storage device of claim 43, wherein the source table further comprises an index.

47. (Original) The program storage device of claim 43, wherein the instructions to retrieve comprise instructions to retrieve the page of memory from a buffer pool.

48. (Original) The program storage device of claim 43, wherein the instructions to retrieve comprise instructions to retrieve the page of memory from a direct access storage device.

49. (Original) The program storage device of claim 43, further comprising instructions to lock the source table before the instructions to delete the identified overflow row from the source table.

50. (Original) The program storage device of claim 49, wherein the instructions to delete comprise instructions to:

identify a constraint associated with the identified overflow row;
disable the identified constraint; and
delete the identified overflow row from the source table.

51. (Original) The program storage device of claim 50, further comprising instructions to lock a table associated with the identified constraint prior to the instructions to disable.

52. (Original) The program storage device of claim 50, wherein the instructions to disable comprise instructions to drop the identified constraint.

53. (Original) The program storage device of claim 50, wherein the instructions to load comprise instructions to:

insert the previously deleted identified overflow row into the source table;
restore the identified constraint; and
unlock the source table.

54. (Original) The program storage device of claim 53, wherein the instructions to unlock the source table comprise instructions to unlock a table associated with the identified constraint.

55. (Original) The program storage device of claim 43, wherein the instructions to unload comprise a structured query language SELECT instruction.

56. (Original) The program storage device of claim 55, wherein the instructions to delete comprise a structured query language DELETE instruction.

57. (Original) The program storage device of claim 55, wherein the instructions to load comprises a structured query language INSERT statement.

58. (Previously Presented) A program storage device, readable by a programmable control device, comprising instructions stored on the program storage device for causing the programmable control device to:

identify an overflow row associated with a source table from a non-source table data source;

unload only the identified overflow row from the source table;

delete the identified overflow row from the source table; and

reload the previously deleted identified overflow row into the source table.

59. (Original) The program storage device of claim 58, wherein the non-source table data source comprises a database log file.

60. (Original) The program storage device of claim 58, wherein the source table further comprises an index.

61. (Original) The program storage device of claim 58, further comprising instructions to lock the source table before the instructions to delete the identified overflow row from the source table.

62. (Original) The program storage device of claim 61, further comprising instructions to unlock the source table after the instructions to reload the identified overflow row.

63. (Original) The program storage device of claim 61, wherein the instructions to delete comprise instructions to:

identify a constraint associated with the identified overflow row;

disable the identified constraint; and

delete the identified overflow row from the source table.

64. (Original) The program storage device of claim 63, wherein the instructions to disable comprise instructions to lock a table associated with the identified constraint prior to the instructions to disable the identified constraint.

65. (Original) The program storage device of claim 64, further comprising instructions to determine if the identified constraint is a deferred constraint and, if it is, not to lock the table associated with the identified constraint.

66. (Original) The program storage device of claim 63, wherein the instructions to disable comprise instructions to delete the identified constraint.

67. (Original) The program storage device of claim 63, wherein the instructions to reload comprise instructions to:

insert the previously unloaded identified overflow rows into the source table;

restore the identified constraint;

unlock the source table.

68. (Original) The program storage device of claim 67, wherein the instructions to restore comprise instructions to rebuild the identified constraint.

69. (Original) The program storage device of claim 58, wherein the instructions to unload, delete and load are performed on a plurality of identified overflow rows at a time.

70. (Original) The program storage device of claim 58, wherein the instructions to unload comprise a structured query language SELECT instruction.

71. (Original) The program storage device of claim 70, wherein the instructions to delete comprise a structured query language DELETE instruction.

72. (Original) The program storage device of claim 71, wherein the instructions to reload comprise a structured query language INSERT instruction.

## IX. EVIDENCE APPENDIX

None.

## X. RELATED PROCEEDINGS APPENDIX

None.